Рубежный контроль №2

Широков Павел ИУ5-21М

Тема: Методы обработки текстов

Решение задачи классификации текстов.

Классификатор 1: LogisticRegression

Классификатор 2: Multinomial Naive Bayes (CNB)

- Для каждого метода необходимо оценить качество классификации
- Сделать вывод о том, какой вариант векторизации признаков в паре с каким классификатором показал лучшее качество.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import numpy as np
import pandas as pd
from typing import Dict, Tuple
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_erro
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
import seaborn as sns
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
import matplotlib.pyplot as plt
from sklearn.svm import SVC, NuSVC, OneClassSVM, SVR, NuSVR, LinearSVR

%matplotlib inline
sns.set(style="ticks")

# !pip install category_encoders
```

```
categories = ["talk.politics.guns", "alt.atheism", "sci.med", "rec.autos"]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

Анализируем датасет и готовим категориальный признак

```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

```
vocabVect = CountVectorizer()
```

```
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 37176
```

```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
thom=33375
morgan=23251
ucs=34360
mun=23527
ca=8754
thomas=33376
clancy=9784
subject=32210
re=28101
```

```
test_features = vocabVect.transform(data)
test_features
```

```
<2214x37176 sparse matrix of type '<class 'numpy.int64'>'
        with 375168 stored elements in Compressed Sparse Row format>
```

```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])
            score = cross_val_score(pipeline1, newsgroups['data'], newsgroups['target'], s
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabulary
classifiers_list = [LogisticRegression(), MultinomialNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Conv
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001
                               '00014': 4, '000152': 5, '000406': 6,
                               '0005111312': 7, '0005111312na3em': 8, '000601': 9,
                               '000710': 10, '000mi': 11, '000miles': 12,
                               '000s': 13, '001': 14, '0010': 15, '001004': 16,
                               '001125': 17, '001319': 18, '001642': 19, '002': 20,
```

```
                            '002142': 21, '002651': 22, '003': 23,
                            '003258u19250': 24, '0033': 25, '003522': 26,
                            '004': 27, '004021809': 28, '004158': 29, ...})
Модель для классификации - LogisticRegression()
Accuracy = 0.951219512195122
============================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001
                            '00014': 4, '000152': 5, '000406': 6,
                            '0005111312': 7, '0005111312na3em': 8, '000601': 9,
                            '000710': 10, '000mi': 11, '000miles': 12,
                            '000s': 13, '001': 14, '0010': 15, '001004': 16,
                            '001125': 17, '001319': 18, '001642': 19, '002': 20,
                            '002142': 21, '002651': 22, '003': 23,
                            '003258u19250': 24, '0033': 25, '003522': 26,
                            '004': 27, '004021809': 28, '004158': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9864498644986449
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001
                            '00014': 4, '000152': 5, '000406': 6,
                            '0005111312': 7, '0005111312na3em': 8, '000601': 9,
                            '000710': 10, '000mi': 11, '000miles': 12,
                            '000s': 13, '001': 14, '0010': 15, '001004': 16,
                            '001125': 17, '001319': 18, '001642': 19, '002': 20,
                            '002142': 21, '002651': 22, '003': 23,
                            '003258u19250': 24, '0033': 25, '003522': 26,
                            '004': 27, '004021809': 28, '004158': 29, ...})
Модель для классификации - LogisticRegression()
Accuracy = 0.971093044263776
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001
                            '00014': 4, '000152': 5, '000406': 6,
                            '0005111312': 7, '0005111312na3em': 8, '000601': 9,
                            '000710': 10, '000mi': 11, '000miles': 12,
                            '000s': 13, '001': 14, '0010': 15, '001004': 16,
                            '001125': 17, '001319': 18, '001642': 19, '002': 20,
                            '002142': 21, '002651': 22, '003': 23,
                            '003258u19250': 24, '0033': 25, '003522': 26,
                            '004': 27, '004021809': 28, '004158': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9774164408310749
============================
```

Лучшая точность была у CountVectorizer с MultinomialNB - 0.986

✓ 35 сек. выполнено в 21:07